process allowing the nodes receiving the pointers to execute the algorithms, the system evaluating the dependency graph with the algorithms passed between the nodes.

21. (TWICE AMENDED) A data structure provided on computer readable storage controlling a computer in association for evaluating a dependency graph of a graphics creation process, the data structure comprising a run time type identification parameter list, a mapping substructure comprising an index mapping, mapping methods, and a data casting matrix, an algorithm pointer, and methods for setting inputs, getting outputs, and evaluating a an algorithm using the passed pointer.

22. (TWICE AMENDED) A method, comprising:

evaluating a dependency graph of a graphics creation process using a computer, comprising:

performing, by a destination node, of an algorithm having a function known to the destination node by evaluating a self evaluating data structure passed from a source node to the destination node using a pointer to the algorithm allowing the destination node to execute the algorithm of the source node and the structure expected to precisely implement the algorithm known to the destination node where the self evaluating data structure can comprise a different algorithm with different parameters and performing the different algorithm actually requested by the destination node.

REMARKS

In the Office Action mailed January 27, 2003 the Examiner noted that claims 1-22 were pending, and rejected all claims. Claims 1-3, 5, 7-9, 11, 15, 17 and 19-22 have been amended, claim 4 has been cancelled and, thus, in view of the forgoing claims 1-3, 5-22 remain pending for reconsideration which is requested. No new matter has been added. The Examiner's rejections and objections are traversed below.

Request For Withdrawal Of Finality

The Examiner is requested to withdraw the finality of this the second and final Office Action. The Examiner has finally rejected claim 21 on pages 2 and 3 of the Office Action. By

finally rejecting the claim 21 the Examiner has removed the right of the applicant to amend claim 21. The phrase of concern to the Examiner has been in the application from the beginning as it was in original claim 21. That is, the Applicant did not make a change to claim 21 that required the Examiner to reject claim 21. The Examiner could have rejected claim 21 in the first Office Action issued in August 2002. The Examiner has introduced a new ground of rejection of claim 21 that was not necessitated by applicant's amendment of claim 21. This is improper under MPEP 706.07(a) and withdrawal of the finality is requested under MPEP 706.07(d).

Objection To Drawings

In the Action on page 2 the Examiner objected to the drawings asserting that the drawings "fail to show computer, keyboard, mouse and display". The Examiner is requested to review figure 3 which shows a computer 12, a keyboard 14, a mouse 16 and a display 18. The Examiners requirement to show these components is respectfully traversed as the drawings do show them. Withdrawal of the objection is requested.

Rejection Of Claim 21

On page 3 of the Action the Examiner rejected claim 21 under 35 U.S.C. section 112, paragraph for failure to be supported by an enabling disclosure. Essentially the Examiner alleges that RTTI is not known. The Examiner is requested to review page 8, lines 5-9 where RTTI is defined as run time type identification which is noted as being a conventional method by which you tag data types and their inheritance relationships for the purposes of identifying an unknown piece of data. The Examiner is also requested to look in any one of a number of computing dictionaries such as http://www.techweb.com/encyclopedia/ or a more general set of dictionaries such as http://www.onelook.com/ to confirm that the term RTTI is well known. It is a fundamental principle of patent law that a patent application need not describe that which is know to those of skill in the art (The specification need not describe the conventional nor disclose what the skilled already possess (see Gen.Elec.Co v. Brenner Comr. Pats,407 F.2d 1253,159 U.S.P.Q. 335,337(Ct.App.D.Col.1968)). Claim 21 has, however, been amended to remove the acronym. Withdrawal of the rejection is requested.

Rejection Of Claims 1-22

On page 3 of the Office Action the Examiner rejected all claims under 35 U.S.C. § 102 as anticipated by Wells.

Wells is directed to a system where a directed acyclic graph (DAG) is used to graphically represent computer analysis operations. The Examiner points to col. 3, lines 13-22 for a teaching of nodes of the graph performing analysis operations. This portion states:

A GDFD is a directed acyclic graph which provides a visual presentation of how data flows through the data analysis system of the present invention. The nodes in the GDFD represent elements in the system, e.g., data sets, analysis operations, and graphs. As with maps generated by graphical programming, the computer generated GDFD of the present invention serves to remind a user what analysis steps have been performed and provides a record of the data processing steps that can be stored, retrieved, and graphically modified to be used in future data analysis sessions.

(see col. 3, lines 13-22)

That is, in Wells each node performs the analysis operations specified for that node. The operations of each node are fixed or static. In contrast, the present invention is directed at using dependency graphs in a graphics creation process in a completely different way. The nodes of the present invention are dynamic. This dynamic nature involves a first node "passing" a second node a "pointer" to an algorithm residing in or associated with the first node. The second node is then able to execute the algorithm of the first node. That is, the nature of the node changes because it can have an algorithm passed to it from another node. There is no discussion in Wells concerning the passing of pointers to algorithms between nodes allowing the destination nodes receiving the pointers to execute the algorithm of the source node. This allows the graph nodes of the present invention to pass algorithms around in the node network. The ability to pass an algorithm from one node to another node allows the graph to essentially dynamically reconfigure itself allowing operations that appears earlier in the graph to be altered. Wells does not recognize the possibility of much less provide this capability. Passing a pointer to an algorithm from one node to another node to allow the node receiving the pointer to execute the algorithm is emphasized in the independent claims (see claims 1, 17, 19, 20, 21 and 22). It is submitted that the present claimed invention patentably distinguishes over Wells and withdrawal of the rejection is requested.

It is submitted that the present claimed invention patentably distinguishes over Wells and withdrawal of the rejection is requested.

The dependent claims depend from the above-discussed independent claims and are patentable over the prior art for the reasons discussed above. The dependent claims also recite

additional features not taught or suggested by the prior art. For example, claim 2 calls for the algorithm to be a self evaluating data structure. Nothing in Wells teaches or suggests this. It is submitted that the dependent claims are independently patentable over the prior art.

No New Issues Raised

Entry Of Amendment Under 37 C.F.R. §1.116 Requested

Applicant requests entry of this Rule 116 response because the finality of the Office Action was premature and should be withdrawn, as discussed above.

Applicant further requests entry because the amendments narrow the issues for appeal claim 4 having been cancelled.

Applicant additionally requests entry because the amendments were not earlier presented because the Applicant believed in good faith that the cited prior art did not disclose the present invention as previously claimed.

Applicant also requests entry of this Rule 116 response because the amendments of claims should not entail any further search by the Examiner since no new features are being added or no new issues are being raised. The claims have been amended to substitute the word "algorithm" for the word "function" to clarify the claimed invention. This equivalence can be found in the title of the application and in numerous places within the specification (see page 1, lines 8-11, page 5, lines 4-6, page 4, lines 20-22). The substitution of an equivalent word for clarity does not raise new issues. The word "algorithm" was in original claim 22 and thus substituting the word in other claims raised no new issues. The claims have also been amended to explicitly recite the use of a pointer to pass an algorithm between nodes. The recitation of a pointer was in original claim 4, now cancelled. Therefore the recitation of a pointer does not raise any new issues.

Applicant requests entry because the amendments do not significantly alter the scope of the claims and place the application at least into a better form for purposes of appeal by reducing the number of claims. No new features or new issues are being raised.

The Manual of Patent Examining Procedures sets forth in Section 714.12 that "any amendment that would place the case either in condition for allowance or in better form for appeal may be entered." Moreover, Section 714.13 sets forth that "the Proposed Amendment should be given sufficient consideration to determine whether the claims are in condition for

allowance and/or whether the issues on appeal are simplified." The Manual of Patent Examining Procedures further articulates that the reason for any non-entry should be explained expressly in the Advisory Action.

Conclusion

It is submitted that the claims satisfy the requirements of 35 U.S.C. 112. It is submitted that the drawings meet the requirements for drawings. It is further submitted that the claims are not taught, disclosed or suggested by the prior art. The claims are therefore in a condition suitable for allowance. An early Notice of Allowance is requested.

If any further fees, other than and except for the issue fee, are necessary with respect to this paper, the U.S.P.T.O. is requested to obtain the same from deposit account number 19-3935.

Respectfully submitted,

STAAS & HALSEY LLP

Date: 4/20/3

By: 

J. Randall Beckers
Registration No. 30,358

700 Eleventh Street, NW, Suite 500
Washington, D.C. 20001
(202) 434-1500

VERSION WITH MARKINGS TO SHOW CHANGES MADE

IN THE CLAIMS:

Please CANCEL claim 4.

1. (TWICE AMENDED) A method, comprising:

evaluating a dependency graph of a graphics creation process using a computer, comprising:

passing a pointer to an algorithm associated with [function of] a first dependency node to a second dependency node allowing the second dependency node to execute the algorithm; and

[evaluating the function] executing the algorithm as part of an evaluation of the second dependency node.

2. (ONCE AMENDED) A method as recited in claim 1, wherein the [function] algorithm comprises a self evaluating data structure.

3. (ONCE AMENDED) A method as recited in claim 2, wherein the [function] algorithm comprises [a function] an algorithm having a defined set and type of inputs and outputs.

4. (CANCELED) A method as recited in claim 2, wherein the structure comprises a function pointer.

5. (ONCE AMENDED) A method as recited in claim 2, wherein the structure comprises [a function] an algorithm calling method.

6. A method as recited in claim 2, wherein the evaluating comprises determining a type of a passed parameter.

7. (ONCE AMENDED) A method as recited in claim 6, wherein the [function] algorithm parameter types are identified dynamically as the dependency graph is executed.

8. (ONCE AMENDED) A method as recited in claim 7, wherein the data structure contains information describing a set of input and output parameters the [function] algorithm accepts.

9. (ONCE AMENDED) A method as recited in claim 8, wherein the information determines if [function] algorithm attribute types within the dependency graph are compatible.

10. A method as recited in claim 9, wherein the data structure comprises default values for all input and output parameters.

11. (ONCE AMENDED) A method as recited in claim 1, further comprising mapping parameters of first and second [functions] algorithms of the first and second nodes.

12. A method as recited in claim 11, wherein said mapping comprises using an index.

13. A method as recited in claim 11, wherein the mapping defines a relationship where input parameters are ignored and output parameters are unmapped and take on default values.

14. A method as recited in claim 11, wherein parameter value and type are passed for the mapping.

15. (ONCE AMENDED) A method as recited in claim 11, wherein the [function] algorithm data structure and value index are passed for the mapping.

16. A method as recited in claim 11, wherein the mapping comprises an index remapping and a matrix of data casting methods which will change one type of data into another.

17. (TWICE AMENDED) A method, comprising:

evaluating a dependency graph of a graphics creation process using a computer, comprising:

passing [a function] a pointer to an algorithm of a first dependency node to a second dependency node allowing the second dependency node to execute the algorithm of the first dependency node, the [function] algorithm comprising a self evaluating data structure comprising [a function] an algorithm calling method and containing information describing a set of input and output parameters the [function] algorithm accepts where the information determines if [function] algorithm attribute types within the dependency graph are compatible and comprising default values for all input and output parameters;

mapping parameters of first and second [functions] algorithms of the first and second nodes, where the mapping comprises an index, defines a relationship where input parameters are ignored and output parameters are unmapped and take on default values, where parameter value and type are passed for the mapping and the [function] algorithm data structure and value index are passed for the mapping; and

executing [evaluating] the [function] algorithm of the first dependency node as part of an evaluation of the second dependency node using the pointer and comprising determining a type of a passed parameter where parameter types are identified dynamically as the dependency graph is executed.

18. A method as recited in claim 17, wherein the mapping comprises an index remapping and a matrix of data casting methods which will change one type of data into another.

19. (TWICE AMENDED) A method, comprising:

evaluating a dependency graph of a graphics creation process

using a computer, comprising:

passing a pointer to an algorithm [a function] from a first node in a node network to a second node in the node network allowing the second node to execute the algorithm; and

[evaluating the function] executing the algorithm as part of an evaluation of the second node.

20. (TWICE AMENDED) An apparatus comprising a computer including a dependency node evaluation system having [functions] pointers to algorithms passed between nodes of a dependency graph of a graphics creating process allowing the nodes receiving the pointers to execute the algorithms, the system evaluating the dependency graph with the [functions] algorithms passed between the nodes.

21. (TWICE AMENDED) A data structure provided on computer readable storage controlling a computer in association for evaluating a dependency graph of a graphics creation process, the data structure comprising [an RTTI] a run time type identification parameter list, a mapping substructure comprising an index mapping, mapping methods, and a data casting matrix, [a function] an algorithm pointer, and methods for setting inputs, getting outputs, and evaluating a [passed function] an algorithm using the passed pointer.

22. (TWICE AMENDED) A method, comprising:

evaluating a dependency graph of a graphics creation process using a computer, comprising:

performing, by a destination node, of an algorithm having a function known to the destination node by evaluating a self evaluating data structure passed from a source node to the destination node using a pointer to the algorithm allowing the destination node to execute the algorithm of the source node and the structure expected to precisely implement the [function] algorithm known to the destination node where the self evaluating

data structure can comprise a different [function] algorithm with different parameters and performing the different [function] algorithm actually requested by the destination node.